EXTRACTING INFORMATION FROM GAMEPLAY VIDEOS USING MACHINE LEARNING TECHNIQUES AND ITS VARIETIES

A Thesis
Presented to
The Academic Faculty

By

Zijin Luo

In Partial Fulfillment
of the Requirements for the Degree
Bachelors of Science in the
School of Interactive Computing

Georgia Institute of Technology

September 2019

EXTRACTING INFORMATION FROM GAMEPLAY VIDEOS USING MACHINE LEARNING TECHNIQUES AND ITS VARIETIES

Approved by:

Dr. Mark Riedl, Advisor
School of Interactive Computing
Georgia Institute of Technology
Date Approved: 9/17/2019
Signature:

Dr. Devi Parikh
School of Interactive Computing
Georgia Institute of Technology
Date Approved: 9/10/2019
Signature: Devi N. Parikh

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1525967. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The ability to extract sequences of game events for high-resolution e-sport games has traditionally required access to the game's engine. This difficulty serves as a barrier to groups who don't possess this access. It is possible to apply deep learning to derive these logs from gameplay video, but it requires computational power that serves as an additional barrier. These groups would benefit from access to these logs, such as small e-sport tournament organizers who could better visualize gameplay to inform both audience and commentators. In this dissertation, we present a combined solution to reduce the required computational resources and time to apply a convolutional neural network (CNN) to extract events from e-sport gameplay videos. This solution consists of techniques to train CNN faster and methods to execute predictions more quickly. These techniques expand the types of machines capable of training and running these models, which in turn extends access to extracting game logs with this approach. We evaluate the approaches in the domain of *DOTA2*, one of the most popular competitive e-sport games. Our results demonstrate our approach outperforms standard backpropagation baselines.

CHAPTER 1

INTRODUCTION

Extracting information from gameplay videos is a critical concern in the game industry. All related fields in this industry can benefit from that information [1]. For examples, real-time data is immensely helpful for game narrators to explain the details of certain events in broadcasting, and game developers can use the feedback information from testing gameplay videos to improve the quality of games. Nevertheless, extracting information demands accurate representations of gameplay events. Before, there are two typical ways to collect such representations: using game logs, or collecting from users. Game logs, the data generated by the game engine, are usually exclusive to game developers and are restricted by game engine design and privacy concerns; information gathered directly from players cannot preserve the data integrity and preciseness. Therefore, we demand a novel method to collect required information - extracting information directly from gameplay videos, which keeps the data precise and avoids most game engine restrictions.

Machine Learning provides one potential solution for extracting necessary information from gameplay videos without restrictions. In the past a few years, many works utilized machine learning techniques to recognize actions and objects from videos. To be specific, there is a piece of work using supervised learning concerned with action recognition for real-world videos [2], which we could apply to the problem of extracting representations of player actions from gameplay videos. Furthermore, as Fulda, Murdoch, and Wingate proposed, unsupervised learning can equally stand a choice to extract players interaction events in realistic games. All of these works prove that machine learning is one of the most practical ways to retrieve data from gameplay videos.

Nevertheless, these machine learning techniques are not perfect: such approaches require massive amounts of paired training data of video labeled with activities, which re-

stricts this approach to those with the access or resources to create a training dataset. While the crucial requirement of computation power is expensive to purchase, the disadvantages stop researchers from applying advanced techniques to high-resolution video games, which usually requires more training data and prolonged time to train a decent production model. Notably, in such cases, the game action recognition model must run in real-time to keep the stability in broadcast in video fame tournaments. These tournaments would likely either lack the high-performance machines associating with deep learning or want to save budgets for computers. Therefore the computation resource requirements and execution time of such a model would matter hugely in determining the success of a machine learning approach.

My research focuses on making the convolutional neural network (CNN) for videos parsing easier to access. CNN is the prominent accurate supervised learning model used to map the data and its label in the computer vision field, and it is easy to modify and extend. For our domain, we employ *Defense of the Ancient 2* or *DOTA2* because it is a popular e-sport and has stylized visuals (meaning that one cannot merely apply real-life event detection models). My goal is to reduce the computational power requirements for training a CNN and execute deep CNNs on low-performance machines. To accomplish them, I split the problem into two sub-problems: training optimization and runtime optimization. In the training optimization, the goal is to reduce the resource and time to train an excellent CNN classifier, and in the runtime optimization, the goal is to decrease barriers to apply pre-trained CNN classifier efficiently in real-time. I choose different complementary machine learning techniques to conquer these two sub-problems. For preparing a CNN faster, I utilize transfer learning and style transfer to reduce the required iterations in the training process. For executing a CNN more quickly, I make use of zero-shot and network pruning technique to reduce the classifier size. After solving both sub-problems with different approaches, the combination of all practical approaches should be a sound solution to reduce the requirement of applying complex CNN classifiers to games. Based on this assumption,

we can derive a system from performing information extraction from some complex and high-resolution video games much more natural. Furthermore, this system is not limited in the video game domain, but it could also be applied in real-life situations like sports broadcasting. In section 5, I will talk about the real-life applications comprehensively.

CHAPTER 2

RELATED WORKS

Player modeling [4], the study of computationally modeling the players' actions in video games, represents a related domain to this paper, given that it requires a representation of player actions as input. Nonetheless, most player modeling systems and methods are typically developed upon the necessary assumption of sufficient computational power and access to comprehensive data of game events. Player modeling research in the domain of commercial games has traditionally required support or partnership with existing game developers [5, 6, 7]. As an alternative, some prior player modeling researches explored the possibility of creating clones of past commercial games [8, 9, 10], but this requires both design and game development knowledge, takes a significant amount of time, and is not tenable for modern, large-scale games. Camilleri, Yannakakis, and Liapis [11] proposed to construct a general model of the player affect across distinct games to make player modeling accessible to more people. However, this approach still requires access to the logging system of each game.

Prior approaches exist to derive approximated game logs from gameplay video. However, the majority of these approaches rely on hand-authored event definitions [12] or a secondary machine vision library like OpenCV [13] and access to all of the images in the game [14, 15]. This latter approach is only viable for two-dimensional games with limited sets of possible models for each game component (called a "sprite").

McCallum, Freitag, and Pereira [16] proposed to use the Maximum Entropy Markov Models to perform information extraction from images inputs, but we choose to focus on the deep learning approaches, which have high potential than HMM, in this paper. Deep neural networks have previously been applied to predict or approximate player experience in a summarizing capacity. For example, approaches applying CNNs to directly predict

player experience metrics like fun and challenge from levels [17] or game logs [18]. Summerville, Guzdial, Mateas, and Riedl [19] derived logs of player movement from gameplay video with OpenCV and then applied a long-short-term memory (LSTM) recurrent neural network to predict player paths through new levels. However, their approach only focused on player movement events. Karavolos, Liapis, and Yannakakis [20] utilized CNNs as a surrogate model of gameplay to predict game level balance based on automated high-level summarizing metrics. Fulda, Murdoch, and Wingate [3] predicted player actions from Skyrim video based on existing machine vision models, but presented inconclusive results. Luo, Guzdial, Liao, and Riedl [1], the work most similar to ours, applied CNNs to convert from gameplay video to representations of player experience for a Super Mario clone, Mega Man, and Skyrim through transfer learning.

Existing prior approaches have improved CNN training time significantly by utilizing explicit input normalization and decreasing the variance. Ioffe and Szegedy [21] introduced the use of batch normalization to minimize changes over each iteration during the training process, which is notorious for slowing down the training by requiring lower learning rates and parameter initialization. Weight normalization [22] is also a solution to simplify the optimization problem and speed up the convergence of stochastic gradient descent during the training process. Another particular method to accelerate training is to use specialized hardware [23]. However, this approach demands additional optimization and support from hardware manufacturers. Those techniques are either general methods to accelerate training or require specialized hardware. They are not designed toward real-world action recognition domains, not for games, and they do not take into account accessibility concerns. We demonstrate that one can use other techniques to accelerate the training process much faster without additional computational power or specialized hardware.

CHAPTER 3

METHODS AND TECHNICAL APPROACHES

The experiment starts with designing and running the standard method to obtain data from gameplay video in *DOTA2*. A standard method stands for a most used way to train a CNN for video game event recognition. The ResNet152 classifier trained by the standard method serves as a baseline by which we compare our attempts to improve training and execution time and to minimize computational power requirements. After obtaining the baseline data, I slightly change the procedure to adapt these complementary machine learning techniques one by one and conclude the results. To train a large-scale model like ResNet152 is additionally a long and resource-intensive task on such a computer. Given sufficient time one could train a ResNet152 model on a computer with a single GPU, but this assumes the ability to dedicate the computer entirely to the training process for hours (depending on the size of the dataset and complexity of the problem). I apply two different approaches to decrease training iteration and time: teacher-student [24, 25] and neural style transfer [26]. The intuition behind both approaches is that retraining an existing ResNet152 model trained on another dataset is faster than training a new ResNet152 model from scratch. After training a CNN, the crucial problem is how to deploy this model and run it in real-time scenarios. The required time to predict one input image or frame is dependent on the size of the model and the computation power and the number of available GPUs. I make use of zero-shot learning and network pruning that focus on shrinking or making use of less of the complete neural network. With running each approach, we record the data and compare it to the baseline.o

Ten most recognizable game events from *DOTA2* are chosen to be the testing task. The reason is that these ten-game events are easily interpretable output from the models, to both expert and casual fans. Further, the focus is on only these most essential events given that

this is an initial investigation into the appropriateness of these approaches in the domain of *DOTA2*. The test split is uniformly drawn from the dataset, to ensure that any variation between the approaches was caused by variations in our approach and not variations in the data.

3.1 Standard Approach

ResNet152 [27] is utilized as the classifier, as it represents a state of the art model on many computer vision problems. *Double Cross Entropy* is chosen as the loss function, and *Adam* is the optimizer. A large and robust gameplay video dataset is usually required in the training processing of ResNet152; thus, for every new domain, a new dataset is collected to support ResNet152 training. The dataset is split into two sections: training dataset and the testing dataset. The training dataset is used to train the CNN, and the testing dataset is utilized to test the performance of trained CNN. *Backpropagation*, the most common way train CNN in deep learning, is utilized for training the ResNet152 model in this standard method.

3.2 Teacher-student Transfer

The teacher-student method is a transfer learning technique that I use in the experiment. The teacher-student method replicates the weights of the pre-trained "teacher" model to a new, generally smaller "student" model. The intuition behind the teacher-student method is that the student model can leverage and modify the features learned by the teacher model on a new task. The procedure is to prepare a "teacher" model on a comprehensive and large dataset and transfer its weight into a smaller "student" model; then, this "student" model is finetuned on a small dataset in the target domain. The advantage of this method is that ones only need to train the teacher model once, and this "teacher" model can be applied to many different "student" models, which require smaller datasets and training times. This approach also ensures that the user does not need to train the teacher model if

one is otherwise available personally. Luo, Guzdial, Liao, and Riedl [1] also demonstrated that a model pre-trained on ImageNet could successfully serve teacher model for gameplay video student models.

3.3 Kernel Neural Style Transfer

Kernel Neural Style Transfer (KNST) is an approach of running style transfer [26] on the learned features of the ResNet152 model trained on ImageNet. This approach is inspired by neural style transfer and generative adversarial networks or GANs, to try to accelerate the training process by updating the trained ResNet152 ImageNet classifier kernels with a designed style. Two independent CNNs are required in this approach: one is the classifier used to extract information, and the other one is the stylizer that selects features from a target gameplay image as the target style for the content of the classifier's first kernel. The prediction accuracy is expected to improve while decreasing training time since the neural networks could converge faster if the classifier already has features from the stylizer.

3.4 Modified Zero-Shot

Zero-shot learning represents a wide set of approaches for deriving classifiers without training or at least without traditionally training on any data [28]. Chao, Changpinyo, Gong, and Sha [29] describes an approach to derive a classifier from a trained neural network by investigating the activations of only a few layers. Mostly the approach looks at the activations over each class in a pre-trained classifier (usually a classifier trained with the standard method, transfer learning, or KSNT), and uses the average activation for each class as a nearest-neighbor classifier, matching a new candidate image to the class with the most similar average activation. This process markedly decreases the GPU usage since it excludes all other layers other than the first few layers. Additionally, the execution speed is accelerated significantly because each neural network forwarding step only goes through these layers instead of full CNN. We anticipated this approach could reduce the execution time

of ResNet152 while still retaining reasonable accuracy.

3.5 Network Pruning: ThiNet

Neural network pruning represents a class of algorithms for removing less important neurons from a typically pre-trained neural network to retain similar performance but with smaller memory and faster processing speed [30]. We employed a particular neural network pruning algorithm referred to as ThiNet [31], to shrink and prune the trained ResNet152 model. ThiNet is an efficient and unified neural network framework that simultaneously accelerates and compresses CNN models in both training and forwarding stages [31]. Different from the original ResNet152 model, ThiNet framework oversees the neurons' weights in the original ResNet152 and only choose the mattering neuron to construct a corresponding ThiNet model. The advantage of applying ThiNet is that it decreases the filters in each convolutional layer while retaining almost the same performance.

CHAPTER 4

RESULTS AND EVALUATIONS

4.1 Teacher-Student Approach Evaluation

In this section, we evaluate the performance of the teacher-student approach in terms of required training time and accuracy. The standard method is applied as a baseline, which trains a ResNet152 model via backpropagation from scratch. The training accuracy and the loss between the teacher-student approach and the standard approach baseline during the training process are compared to conclude results. Figure 4.1 and Figure 4.2 represent the training accuracy and loss trend respectively over first 20 iterations for both approaches. The red line represents the teacher-student model, and the standard method is in blue.

Both figures demonstrate the teacher-student approach converges faster in comparison to the standard method. More specifically, the teacher-student model can reach near 100% training accuracy after five iterations, with the loss approaching zero and remaining steady. However, the standard method is still in the converging stage after twenty iterations, and both accuracy and loss curves have fluctuations. Note that the standard method took roughly fifty iterations to fully converge, representing an improvement of approximately an order of magnitude. In terms of final testing accuracy, the teacher-student approach achieves 99.8% test accuracy after converging, compared to 94.6% for the standard method. While this difference may seem small, over a second in a live stream, it would be the difference between an average of no errors and three errors. Further, our primary focus is in reducing training time, which teacher-student accomplishes compared to the standard method.

The results demonstrate that the teacher-student transfer learning approach is both more efficient and more accurate in comparison to the standard method. Based on the empiri-

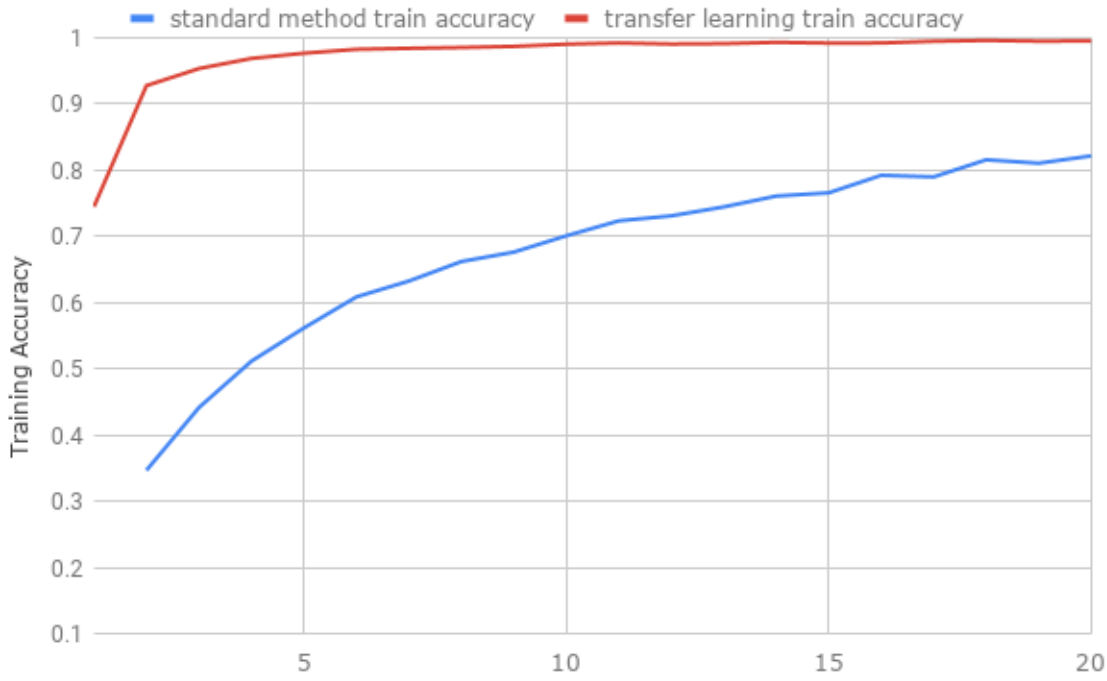

Figure 4.1: The training accuracy over the first 20 iterations for the teacher-student approach and standard method baseline

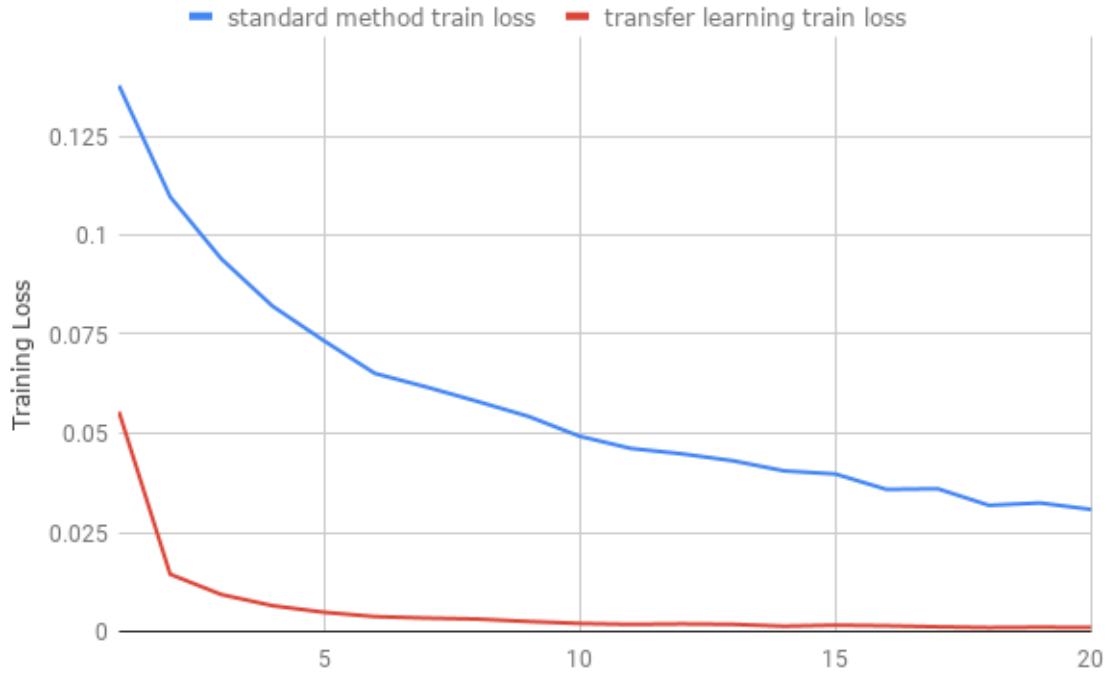

Figure 4.2: The training loss over the first 20 iterations for the teacher-student approach and standard method baseline

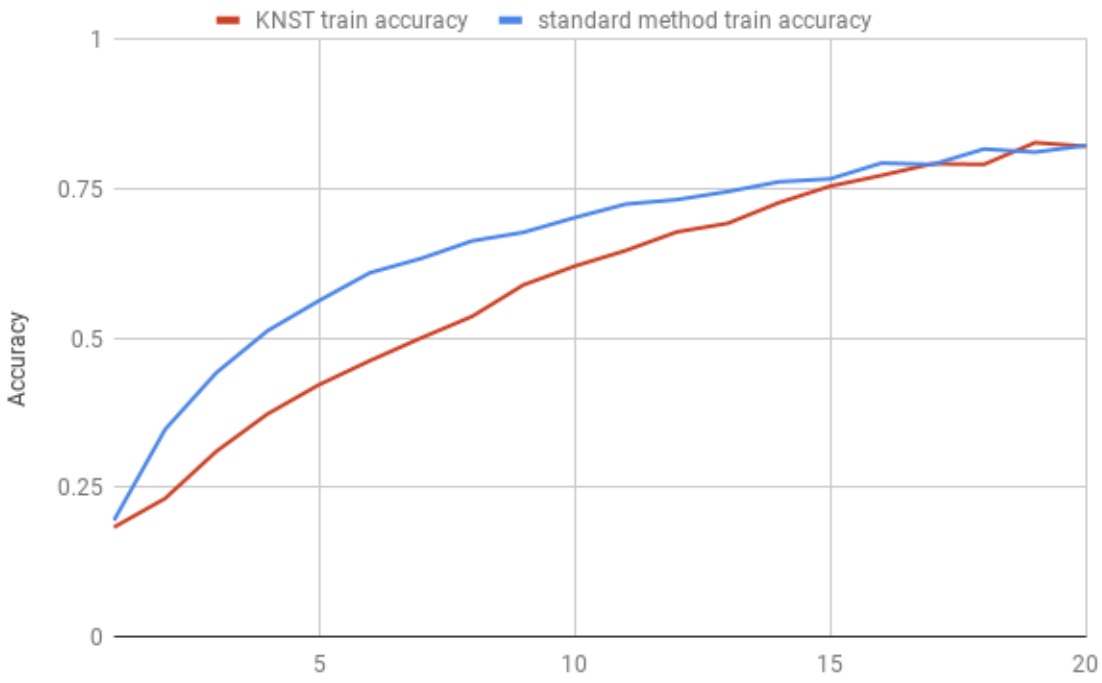

Figure 4.3: The training accuracy over the first 20 iterations for KNST and standard approach baseline.

cal data collected, applying the teacher-student approach takes on;y 10% of the required training epochs of the standard method to converge. This result indicates an impressive improvement in terms of accessibility for those with time constraints, especially those without high-end hardware.

4.2 Kernel Neural Style Transfer Evaluation

In this section, we evaluated KNST using the same method as in Section 4.1. If we see a similar or more considerable improvement utilizing KNST as the teacher-student approach in comparison to the standard method that would lend evidence to the importance of aesthetic differences between real-world images and games with a *DOTA2*-like visual aesthetic.

The results are demonstrated in Figure 4.3 with the training accuracy and Figure 4.4, which includes the training loss over the first 20 iterations. The training loss does not

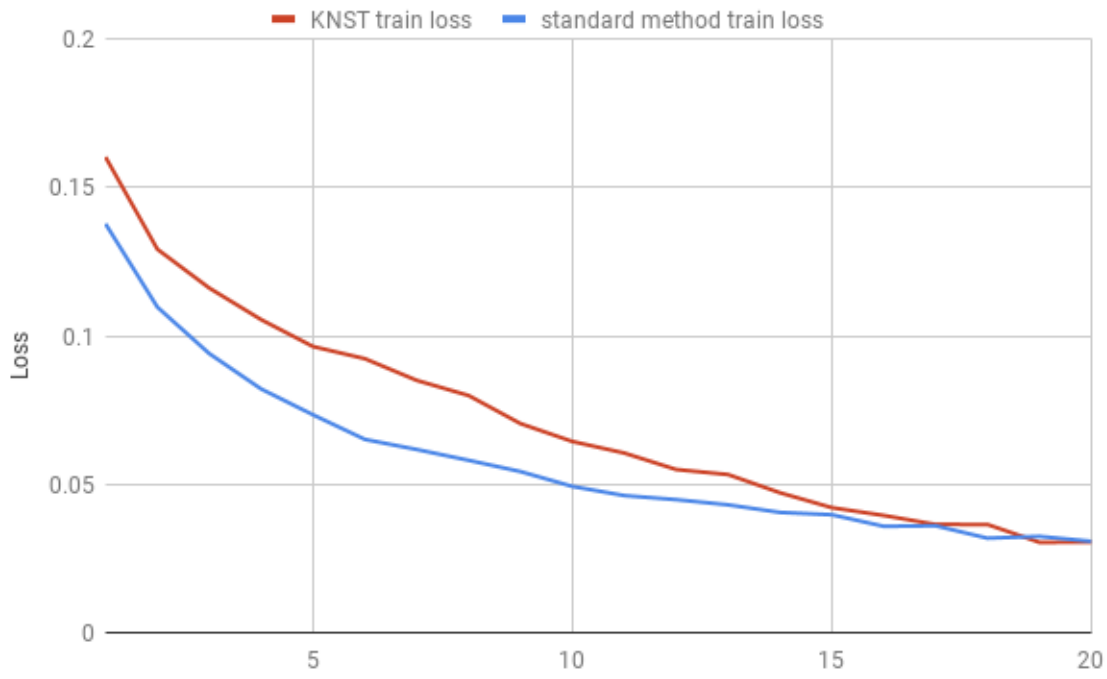

Figure 4.4: The training loss over the first 20 iterations for KNST and standard approach baseline

converge faster than the standard method, and the training accuracy has no noticeable improvements. Markedly, it's worthy to note that KNST even appears to converge somewhat slower than the standard method during the initial iterations.

These results demonstrate that there is no apparent improvement from the KNST method compared to the standard method. Further, to stylize the first layer of the classifier model requires additional GPU memory, meaning that each training step for the KNST approach is longer than a single training step for the standard method. However, the final test accuracy of the KNST method was roughly two percent higher than the standard method test accuracy. This result indicates that while KNST may not have outperformed the standard method in terms of training time or accuracy, it may have lead to more general final features.

This experiment had several limitations. We only used one image in the stylizer to stylizing the first kernel in the classifier. It is quite likely that this image cannot represent

all features that appear in our dataset. Further, KNST mostly had to train two models simultaneously. Hence, it's reasonable that KNST didn't perform well in training process acceleration.

Based on these results, we have to admit that KNST is not a mature approach to accelerate the training process. However, this method could be proper consideration for alternative classification tasks. For example, if the game's visual aesthetics were significantly less like visual aesthetics.

4.3 Modified Zero-Shot Approach Evaluation

In section 3.4, we demonstrate the motivation and structure of this modified zero-shot approach. In this section, we evaluate the execution time of this modified zero-shot approach on *DOTA2* dataset thoroughly against the standard approach. Ideally, this approach will improve in execution time without a significant loss in accuracy.

We include the average test accuracy in Table 4.1. The result shows that our expectation was not met in terms of a non-significant loss of accuracy, with all variations performing with about half the accuracy of the complete ResNet152. Two variations are included in our results beyond the "Only First 4 Layers" version described in the previous section. The first variation adds the layer1 module, which has a total of 13 layers, and the second adds both the layer1 and layer2 modules, for a total of 37 layers. It's explicit that there is a considerable improvement in the 13-layer and 37-layer experiments compared to the first-4-layer experiment: with the 13-layer classifier reaching about 51% test accuracy and 37-layer classifier reaching 56% test accuracy. However, even though these two new classifiers outperform the first-4-layer classifier; the accuracy they reach is insufficient for any real-world scenarios. Despite this we did note our predicted improvement in speed, with the t time (s) value of Table 4.1 referring to the time it took to process the complete *DOTA2* dataset, cutting our execution time to a third compared to the full ResNet152 model. Across the variations, we found out that the runtime does not increase significantly as additional

Table 4.1: A comparison of different number layers in Zero-Shot Learning.

	Only First 4 Layers	After Layer1 Module	After Layer2 Module
Average Accuracy	42.43%	50.99%	56.25%
Time (s)	31.002	32.586	34.113

Table 4.2: A comparison of different ThiNets and original ResNet152.

	ThiNet30	ThiNet50	ThiNet70	ResNet152
Parameters	6573806	10287296	14847686	57992384
Memory Usage (MB)	1405	1473	1541	2669
Parameter-Memory Ratio	4678.9	6983.9	9635.1	21728.1
Execution Time (s)	23.665	23.564	23.984	24.516
Test Accuracy	99.7%	99.8%	99.8%	94.6%

layers are added. However, it appears we would quickly plateau by increasing the size of the zero-shot classifier, without achieving the level of accuracy we'd need to deploy this approach in a real-world environment.

In this section, the result demonstrated that using this modified zero-shot approach can reduce the GPU memory usage and execution time while sacrificing prediction accuracy. This tradeoff is too harsh to deploy in our target scenario, real-time e-sports streams on a single GPU computer. However, depending on the scenario, such a loss of accuracy may be acceptable due to the size of datasets that need to be processed, an initial processing step in a longer pipeline.

4.4 Network Pruning Evaluation

In this section, we evaluate the different ThiNet models and ResNet-152 in terms of the number of parameters, GPU Memory Usage, Execution time, and test accuracy. Shown in Table 4.2, the GPU memory usage of each CNN increases when the parameters of each

CNN increases. Also, as the number of CNN's parameters increase, the parameter-memory ratio increases, which implies that increasing the number of parameters results in requiring less memory per parameter on average. We also note that the execution time for all four models, this time over only the test frames of our dataset was almost the same. To the best of our knowledge, this result is due to the optimization of cudnn, and that forwarding is not the most costly process. The initializing processes of different models also have almost the same computation cost. Thus the particular per-frame processing time is lower, but there is some start-up time for each model, and this is mostly unchanged.

We show this approach can effectively reduce the parameters of CNNs, but due to the optimization of low-level code compilation, the differences in execution time are not distinct. Compared to the modified zero-shot approach we proposed in the previous section, ThiNet is a better solution to reduce the GPU memory usage while keeping high prediction accuracy. The cause is that, in our experiment, all ThiNet models have 152 layers, and they can fit the dataset much better than the modified zero-shot approach. Besides, there are significant savings in terms of memory usage, dropping to nearly half of the complete ResNet152 model.

CHAPTER 5
DISCUSSION

Our work proves that a combined solution of training acceleration and running optimization is one suitable solution to reduce the requirements to apply accurate CNNs to game event prediction from e-sport gameplay video. However, the teacher-student approach requires a teacher model pre-trained on a large, high-quality dataset. While there are many such pre-trained models available, they cannot cover all possible scenarios, and we identify constructing a high-quality dataset and training an initial teacher model as future problems. For example, consider attempting to apply this approach to a retro game with a pixelated aesthetic. One would expect, and we have found from initial tests, that real-world datasets do not transfer nearly as well to this aesthetic as the cartoony, but high quality aesthetic of *DOTA2*. Thus, to apply the teacher-student method, one would need an existing, high-quality dataset with a pixelated aesthetic. While there have been prior attempts at this process [1], they are not of sufficient quality for real-world applications. The kernel neural style transfer approach is not successful, but it represents an innovative strategy and may be helpful in extreme cases like this. In future work, we hope to address these issues to make these approaches accessible to even wider audiences.

While we anticipate that these approaches should extend to games with semi-realistic aesthetics at least as close as *DOTA2*, we acknowledge we would likely need to make changes to our approach in certain situations. For example, first-person shooting e-sports game such as *Overwatch*, which does not have the same over-the-top view as *DOTA2* has more events that may occur off-screen. However other e-sport games like *Hearthstone* and *Artifact*, which are card games with a good deal of hidden information, may still be successful domains for this approach given that all cards are still visible once played.

Besides applying these techniques and similar ones for gaming event extraction, we

also look forward to extending the usage to non-gaming events in the real world, like sports broadcasting. Nowadays, video games are more and more realistic, and there are a lot of common traits shared by e-sports games and real-life sports. Based on judgments, we can surely imply that these machine learning techniques could work in real-world situations.

We acknowledge that a current limitation of this work is that we do not attempt to run any of these approaches in the context of our goal scenario. While we note that this was meant primarily as a motivating example for the kind of applications one might imagine in the future from these technologies, we still anticipate a need to apply these methods in the real world. This also underscores a limitation in our focus primarily on hardware accessibility, issues related to financial and technical knowledge accessibility undoubtedly arise in a real-world application. We anticipate the ability to address the technical knowledge barriers with explainable AI and other human-computer interaction methods, which open further avenues for future research.

CHAPTER 6

CONCLUSION

Our work proposed a combined solution to reduce the requirements for extracting information from the e-sport *DOTA2* gameplay videos: specifically, how to decrease CNN training time, execution time, and memory requirements. Since all four approaches are independent, the performance of the combination of methods would be an assembly of the performance of all methods. We investigate the ability and performance of utilizing teacher-student transfer learning on *ImageNet* and kernel neural style transfer of *DOTA2* to train a CNN faster. Then, we demonstrate the impact of zero-shot prediction and neural network pruning on CNN execution time and memory usage. We evaluate these approaches against the standard backpropagation method. Further, we present a corpus we developed for *DOTA2* event recognition. Our results demonstrate that both approaches can help reduce the requirement of applying complex CNN classifiers to e-sport games and a combined solution of these two approaches has a high potential for minimizing technical barriers to the accessibility of these approaches. After all, we discussed the limitations of current techniques and potential applications to real-world activities in the future.

REFERENCES

[1] Z. Luo, M. Guzdial, N. Liao, and M. Riedl, "Player experience extraction from gameplay video," *arXiv preprint arXiv:1809.06201*, 2018.

[2] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[3] N. Fulda, B. Murdoch, and D. Wingate, "Threat, explore, barter, puzzle: A semantically-informed algorithm for extracting interaction modes," in *Proceedings of the 1st Knowledge Extraction from Games Workshop*, AAAI, 2018.

[4] G. N. Yannakakis, P. Spronck, D. Loiacono, and E. André, "Player modeling," in *Dagstuhl Follow-Ups*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, vol. 6, 2013.

[5] A. Drachen, A. Canossa, and G. N. Yannakakis, "Player modeling using self-organization in tomb raider: Underworld," in *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, IEEE, 2009, pp. 1–8.

[6] A Sabik and R Bhattacharya, "Data-driven recommendation systems for multiplayer online battle arenas," PhD thesis, Masters thesis, Johns Hopkins University, 2015.

[7] S. Makarovych, A. Canossa, J. Togelius, and A. Drachen, "Like a dna string: Sequence-based player profiling in tom clancys the division," in *Artificial Intelligence and Interactive Digital Entertainment Conference*, York, 2018.

[8] J. Togelius, S. Karakovskiy, and R. Baumgarten, "The 2009 mario ai competition," in *IEEE Congress on Evolutionary Computation*, IEEE, 2010, pp. 1–8.

[9] N. Shaker, M. Shaker, and M. Abou-Zleikha, "Towards generic models of player experience," in *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2015.

[10] K. Siu, M. Guzdial, and M. O. Riedl, "Evaluating singleplayer and multiplayer in human computation games," in *Proceedings of the 12th International Conference on the Foundations of Digital Games*, ACM, 2017, p. 34.

[11] E. Camilleri, G. N. Yannakakis, and A. Liapis, "Towards general models of player affect," in *Affective Computing and Intelligent Interaction (ACII), 2017 Seventh International Conference on*, IEEE, 2017, pp. 333–339.

[12] L. B. Jacob, T. C. Kohwalter, A. F. Machado, E. W. Clua, and D. de Oliveira, "A non-intrusive approach for 2d platform game design analysis based on provenance data extracted from game streaming," in *Computer Games and Digital Entertainment (SBGAMES), 2014 Brazilian Symposium on*, IEEE, 2014, pp. 41–50.

[13] G. Bradski and A. Kaehler, "Opencv," *Dr. Dobbs journal of software tools*, vol. 3, 2000.

[14] M. Guzdial and M. Riedl, "Game level generation from gameplay videos," in *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2016.

[15] L. Bao, J. Li, Z. Xing, X. Wang, X. Xia, and B. Zhou, "Extracting and analyzing time-series hci data from screen-captured task videos," *Empirical Software Engineering*, vol. 22, no. 1, pp. 134–174, 2017.

[16] A. McCallum, D. Freitag, and F. C. Pereira, "Maximum entropy markov models for information extraction and segmentation.," in *Icml*, vol. 17, 2000, pp. 591–598.

[17] M. Guzdial, N. Sturtevant, and B. Li, "Deep static and dynamic level analysis: A study on infinite mario," in *Experimental AI in Games Workshop*, vol. 3, 2016.

[18] N. Liao, M. Guzdial, and M. Riedl, "Deep convolutional player modeling on log and level data," in *Proceedings of the 12th International Conference on the Foundations of Digital Games*, ACM, 2017, p. 41.

[19] A. Summerville, M. Guzdial, M. Mateas, and M. O. Riedl, "Learning player tailored content from observation: Platformer level generation from video traces using lstms," in *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2016.

[20] D. Karavolos, A. Liapis, and G. N. Yannakakis, "Using a surrogate model of gameplay for automated level design," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, IEEE, 2018, pp. 1–8.

[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[22] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 901–909.

[23] K. Ovtcharov, O. Ruwase, J.-Y. Kim, J. Fowers, K. Strauss, and E. S. Chung, "Accelerating deep convolutional neural networks using specialized hardware," *Microsoft Research Whitepaper*, vol. 2, no. 11, 2015.

[24] J. H. Wong and M. J. Gales, "Sequence student-teacher training of deep neural networks," 2016.

[25] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," *arXiv preprint arXiv:1805.04770*, 2018.

[26] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2414–2423.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[28] B. Romera-Paredes and P. Torr, "An embarrassingly simple approach to zero-shot learning," in *International Conference on Machine Learning*, 2015, pp. 2152–2161.

[29] W.-L. Chao, S. Changpinyo, B. Gong, and F. Sha, "An empirical study and analysis of generalized zero-shot learning for object recognition in the wild," in *European Conference on Computer Vision*, Springer, 2016, pp. 52–68.

[30] R. Reed, "Pruning algorithms-a survey," *IEEE transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.

[31] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," *arXiv preprint arXiv:1707.06342*, 2017.